Article

A Jamming-Resilient and Scalable Broadcasting Algorithm for Multiple Access Channel Networks

Bader A. Aldawsari * and Jafar Haadi Jafarian

Department of Computer Science and Engineering, College of Engineering, Design and Computing,
University of Colorado Denver, Denver, CO 80204, USA; haadi.jafarian@ucdenver.edu

* Correspondence: bader.aladowsari@ucdenver.edu

Abstract: Multiple access channel (MAC) networks use a broadcasting algorithm called the Binary Exponential Backoff (BEB) to mediate access to the shared communication channel by competing nodes and resolve their collisions. While the BEB achieves fair throughput and average packet latency in jamming-free environments and relatively small networks, its performance noticeably degrades when the network is exposed to jamming or its size increases. This paper presents an alternative broadcasting algorithm called the K-tuple Full Withholding (KTFW), which significantly increases MAC networks' resilience to jamming attacks and network growth. Through simulation, we compare the KTFW with both the BEB and the Queue Backoff (QB), an efficient and high-throughput broadcasting algorithm. We compare the three approaches against two different traffic injection models, each approximating a different environment type. Our results show that the KTFW achieves higher throughput and lower average packet latency against jamming attacks than both the BEB and the QB algorithms. The results also show that the KTFW outperforms the BEB for larger networks with or without jamming.

Keywords: jamming; wireless network; broadcast algorithm; multiple access channels; broadcast protocol

Citation: Aldawsari, B.A.; Jafarian, J.H. A Jamming-Resilient and Scalable Broadcasting Algorithm for Multiple Access Channel Networks. *Appl. Sci.* **2021**, *11*, 1156. <https://doi.org/10.3390/app11031156>

Received: 24 December 2020

Accepted: 23 January 2021

Published: 27 January 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Ethernet and wireless local area networks (WLANs) have become the main building blocks of TCP/IP networks. These networks use multiple access channel (MAC) protocols to enable multiple nodes to communicate fairly and robustly through a shared channel. Ethernet networks use carrier sense multiple access/collision detection (CSMA/CD) in which the sender stops the transmission of the packet when a collision is detected. In addition, WLANs use carrier sense multiple access/collision avoidance (CSMA/CA), where the nodes attempt to avoid collision by waiting for an inter-frame spacing (IFS) time before transmitting a packet. These MAC protocols rely on an algorithm called binary exponential backoff (BEB) to mediate access to the channel.

One of the main challenges in these MAC protocols is their vulnerability to signal jamming attacks [1]. This vulnerability emanates from their shared-access nature, which allows malicious adversaries to bottleneck the channel using off-the-shelf hardware [2]. Other types of networks that use a different topology, such as mesh topology, require more jamming resources than MAC networks. These networks use more than one shared medium, which complicates the jamming attack [3,4].

A malicious node can simply jam the network via signal transmission on the shared channel. This jamming attack causes transmitted network packets to be garbled which prevents communication between the nodes. While the BEB algorithm achieves reasonable throughput and average latency in non-adversarial settings, its performance degrades when the shared medium is exposed to signal jamming attacks, especially when the number of nodes on the shared channel is high. Many researchers have focused on

devising methods to improve the jamming resistance of MAC networks. Some of these approaches are ineffective, which only works for low jamming rates and reduces the network's transmission rate. Other methods can cause significant packet delays, such as channel hopping [5–9].

WLANs also suffer from severe performance bottlenecks when a huge number of nodes contend for access to the channel [10–13]. The throughput of the network degrades significantly for large networks. This degradation affects communications, especially in places with a typically large number of nodes such as conference venues and public hotspots. The broadcast algorithm is one of the main factors affecting the performance of WLANs. Other problems impairing WLANs performance include intercell interference and network traffic overhead caused by the change in AP association [14,15]. In this work, we attempt to address one of the performance issues by proposing a new jamming-resilient broadcast algorithm that is more efficient than the standard BEB algorithm.

This paper explores a different direction: improving MAC network performance and resilience to jamming attacks by replacing the BEB algorithm with a more jamming-resistant broadcast algorithm. To this aim, we propose a new broadcast algorithm called the K-tuple full withholding (KTFW) algorithm. The BEB algorithm relies on randomness to avoid collision of retransmission packets, whereas our algorithm manages access to the channel in a deterministic manner. Although the KTFW algorithm is deterministic, it does not require a central node, and the algorithm is executed in a distributed manner by all the nodes. The KTFW algorithm is designed specifically for MACs with collision detection capability. Although existing WLANs do not have collision detection, research has been done on providing collision detection capability for WLANs [16–18].

We model the attack as a random (also called memoryless) jammer that corrupts transmitted packets with a certain jamming rate between 0 and 1, where 0 means no jamming and 1 represents continuous jamming. Using simulations, we compare the KTFW with both the standard BEB and the queue backoff (QB) [19], a high-throughput broadcast algorithm, and measure their performances in terms of *throughput* and *average packet latency* under various traffic injection models, jamming rates, and network sizes. We quantify these performance metrics for two different adversarial models of packet injection: the leaky bucket injection model (LBIM) and the randomized injection model (RIM) [20,21]. Through experiments with real network traffic and using a modified KL-divergence metric, we show that the considered models provide an accurate and inclusive approximation of real-world network traffic.

Our evaluation shows that the KTFW algorithm makes MAC networks significantly more resilient to jamming attacks than the BEB and the QB algorithms. Under jamming, the KTFW achieves higher throughput compared to both the BEB and the QB. As the network size increases, the KTFW's advantage over the BEB increases even further. This superiority is because the throughput of the BEB degrades as the network size increases while the KTFW maintains a high throughput regardless. For a network with 100 nodes, the KTFW can achieve up to 50% more throughput than the BEB for certain jamming rates. The KTFW algorithm has a significant and consistent superiority over both the BEB and QB algorithms in terms of the average packet latency and achieves much lower average packet latencies for all jamming rates and network sizes. Interestingly, our results reveal that, even with no jamming, the KTFW outperforms the BEB algorithm in terms of throughput and latency for larger network sizes. Although the KTFW algorithm requires the node clocks to be synchronized, it is mitigated through a synchronization step executed once by new connected nodes.

The rest of the paper is organized as follows. In Section 2, we review the related works. Then, in Section 3 we describe some of the technical preliminaries. Afterward, we present the new proposed broadcast algorithm in Section 4, and then we define the traffic injection model in Section 5. Next, the jamming model is detailed in Section 6 and the evaluation is discussed in Section 7. Section 8 concludes the paper.

2. Related Works

Many studies have investigated the security of MAC networks and their susceptibility to cyber-attacks [1,22,23]. In general, network attacks can be classified into passive attacks and active attacks [24]. In this work, the focus is on active attacks in which an attacker attempts to disrupt a network through signal jamming to prevent communications.

Jamming attacks are a type of denial of service (DoS) attack on the physical layer [25]. A jammer tries to disrupt a communication channel by sending a radio signal or by random transmission on the channel [26]. These jamming attacks causes the transmission of packets to fail which blocks communication between legitimate nodes.

In this paper, we focus on a generalized random jamming model where a jammer is transmitting a jamming pulse rate $j \in [0, 1]$, where j denotes the number of jamming signals that the jammer creates every second [27]. For any time interval t , the probability of generating a jamming signal is $1 - e^{-jt}$.

In addition to random jamming, other jamming models have been proposed in the literature [26,28]. A constant jammer is a well-resourced adversary that uses a strong signal to completely block off all communications on the channel [25]. This is equivalent to a random jammer with a jamming rate $j = 1$. With constant jamming, as no packet could ever be transmitted, the broadcast algorithm becomes irrelevant, and thus threat model falls outside the scope of this work.

A deceptive jammer continuously transmits packets that appear to be legitimate without following the underlying protocol [28]. As a result of this continuous transmission, all other nodes remain in a “receiving state”, which causes a disruption to the network activity. Again, a deceptive jammer is equivalent to a random jammer with a jamming rate of $j = 1$ and falls outside the scope of this work.

An intelligent jammer conducts jamming by cautiously attacking specific control or data packets in order to disrupt the communication with minimal noise, thus avoiding detection [29]. This type is classified into four categories: CTS corruption jamming, ACK corruption jamming, DATA corruption jamming, and DIFS wait jamming. As the jamming rate is lower than 1, the broadcast algorithm should have a positive impact on resilience against intelligent jammers. However, in this work, we only focus on random jamming and leave evaluation against intelligent jammers to future work.

Several techniques have been proposed in the literature to mitigate jamming attacks. One of the strategies against jamming attacks on wireless networks is the use of channel reassignment. In this approach, nodes would switch to another channel at regular intervals or when jamming attacks are detected [30]. This strategy is known as channel hopping or channel surfing [5,6] and is especially effective against constant jammers. Channel reassignment and traffic rerouting have also been studied in multi-hop networks [31]. The timing channel is another strategy that allows for low-rate communication in the presence of jamming attacks [7].

Next, the related work for other types of jamming mitigation techniques at the physical layer is presented. Dou et al. [32] proposed an anti-jamming method for internet of things (IoT) networks. Their approach uses a spreading-time technique for anti-jamming, which is used as part of a novel automatic control allocation model. They showed the effectiveness of the proposed method through simulations. Other anti-jamming techniques have been explored for IoT networks, such as the work by Tang et al. [33]. They studied and proposed an anti-jamming technique that exploits the weakness of a deceptive jammer. Their method involves adjusting the transmission power by legitimate nodes to avoid being detected by the deceptive jammer. Tseng et al. [34] developed a novel anti-jamming resource allocation algorithm for single-input multi-output (SIMO) OFDMA video transmission systems. The anti-jamming method considers the angle between the jammer and the sender channel's vector. The simulation results showed improvements in the peak signal-to-noise ratio.

Lin and Noubir [1] studied the ability of WLANs to withstand network attacks. For the IEEE 802.11 and IEEE802.11b packets, garbling a single bit of a packet will render the packet useless. They proposed an anti-jamming technique using both cryptographically

strong interleavers and error-correction codes. This anti-jamming method can maintain a good throughput under moderate jamming when using LDPC codes.

Next, we present an overview of broadcast algorithms for MAC networks. The broadcast algorithm design can be classified into three main types: ALOHA, splitting algorithms, and window-based algorithms. The first broadcast algorithm, known as ALOHA, was developed in the 1980s [35]. It works by having stations reattempt transmission of collided messages with a constant probability. Slotted ALOHA was developed, which improves the maximum throughput by partitioning time slots [36]. The station can only transmit messages at the beginning of a time slot. Tree-based algorithms were developed individually and independently by Capetanakis [37], Hayes [38], and Tsybakov and Mikhailov [39]. They implemented a broadcast algorithm based on the idea of a binary search tree. In this algorithm, only nodes that were involved in collision participate in the collision resolution. It works by splitting the set of nodes into two halves by flipping a coin. This process continues until a set with a single node is found.

Anantharamu and Chlebus developed several broadcast algorithms for MAC networks, including Counting Backoff, Queue Backoff, and Quadruple Round [19]. These algorithms are deterministic and do not use any randomization. The authors studied them using an adversarial injection model with a restriction on the number of stations that can be activated. They showed that the algorithms could attain bounded packet latency for different injection rates.

3. Technical Preliminaries

In this work, we consider MAC networks where the nodes communicate through a single shared medium/channel. This shared medium can be Ethernet (LANs) or radio frequency (WLANs). The nodes in a MAC network communicate through the broadcasting of packets on the shared channel. Successfully transmitted packets are received by all nodes on the shared channel. The broadcast algorithms are studied under synchronous communication, in which time is partitioned into equal slots, referred to as rounds. The locally synchronous model is assumed, where each node has its own local clock that ticks at the same rate. Therefore, the nodes do not have knowledge of the global round number while they keep track of local rounds. Packets are assumed to have a fixed size and are transmitted at the start of a round. At most one packet can be transmitted in a single round. The process of transmitting or receiving a packet takes exactly one round. Thus, in a round, a node can either broadcast a packet or listen to feedback from the channel. A node can transmit at most one packet from its queue, which will be encapsulated in a message. The channel is assumed to have collision detection such that nodes can identify when a collision has taken place. The feedback from the channel exists in three types: packet, collision, and silence. When two or more nodes transmit a packet in the same round, a collision will be heard on the channel. Otherwise, a packet will be heard as feedback when exactly one node transmits on the channel. The feedback from the channel is silence when no node is transmitting any packets. A node attempts to broadcast all packets in its queue through the execution of the broadcasting algorithm.

A node can be in one of two states: passive or active. All nodes begin in the passive state, with no packets in their queues. A node is considered active when it has one or more packets in its queue. Thus, a passive node is activated when it is injected with at least one packet. When all packets in a node's queue have been successfully transmitted, the node becomes passive. It is assumed that packets are never dropped such that all nodes attempt to retransmit a packet that has experienced a collision until it is eventually heard on the channel.

Next, two broadcast algorithms used for evaluating the performance of the proposed broadcast algorithm are described. One of the algorithms is the QB algorithm, a high-performance state-of-the-art broadcast algorithm [19]. It was developed for the category of adaptive activation-based algorithms and works similarly to queue-based methodology, where nodes are processed for broadcasting based on a "first come, first served" paradigm.

The QB algorithm operates by having a node transmit a packet immediately in the next round. If there are no other active nodes, a packet will be transmitted successfully in the next round, and the node will record its position in the queue. When a collision occurs after transmitting a packet, the node continues counting the number of collisions since its activation. The leading node keeps attaching the queue size to each transmitted packet, which in conjunction with the number of collisions, allows each node to keep track of its position in the queue. An over bit flag is then attached to the final packet in the queue, which informs other nodes about the termination of the current leading node in the queue. The next node in the queue begins transmitting packets from its queue when an over bit flag is detected.

The other algorithm used in measuring performance is the BEB algorithm, which is the standard algorithm implemented as part of the MAC protocols in IEEE 802.11b [40]. This is a randomized algorithm implemented as part of the MAC protocol in the IEEE 802.11 standard. In backoff algorithms, when a packet experiences a collision, the node waits for a random duration of time before retransmitting the packet. The random waiting is chosen from a window range that is determined based on the number of collisions and the backoff function: $W_i = W_0 \cdot 2^i$ where W_0 is the initial window size and i is the backoff stage number. The parameter W_0 specifies the minimum window range from which the waiting time is selected, and it is set to a predefined value in the IEEE 802.11 standard.

4. Proposed Broadcast Algorithm

In this section, we describe the new proposed broadcast algorithm known as the KTFW algorithm. The KTFW algorithm is designed for the category of non-adaptive, full-sensing algorithms [19]. A broadcast algorithm is non-adaptive when no node can attach additional information to packets, and it is full-sensing when a node continues listening to feedback from the channel at all times. The new proposed algorithm is distributed because each node executes the broadcast algorithm deterministically based on the feedback from the channel. The algorithm does not use randomization and does not require a central node; see the pseudocode for the algorithm in Algorithm 1. Although the algorithm's specification is described for synchronous communication, this does not impose any restriction on its design. The algorithm can be implemented in asynchronous communication, even without requiring synchronized clocks. This limitation is overcome through a simple yet effective solution. The solution is a *synchronization* step that applies only to new nodes joining the network, allowing them to synchronize with existing nodes. This synchronization step avoids the necessity for global synchronization and enables both existing and new nodes to communicate on the same network. Therefore, the KTFW can be implemented and used in practical network scenarios. The synchronization step is described further in this section.

The main idea of the KTFW algorithm is to divide the execution into segments consisting of k rounds, and a segment terminates when all activated nodes of the segment have successfully transmitted all their packets. Although the algorithm looks similar to existing tree-based algorithms, it works differently from them as it uses a novel approach in handling collisions and deciding which nodes participate in the broadcast. For each segment, a binary search is repeated to process the packets until all packets in all nodes have been transmitted successfully. An example of how the binary search works is presented in Figure 1, in which the search starts from top to bottom and from left to right. Although the KTFW algorithm resembles tree-based broadcast algorithms, it uses novel approaches such as *withholding* the channel and it does not rely on randomization.

Next, we describe how the KTFW algorithm works at the beginning of a segment. All nodes that are activated during a segment enter a segment processing phase, in which each node is allowed to transmit its packets on the channel. In a segment, each node is identified by its activation round, such that all activated nodes are labeled with consecutive IDs starting from 1. In this work, the injection model is 1-activating such that a maximum of one node can be activated in a single round. Therefore, in a segment of k rounds, a maximum of k nodes can be activated, and no two nodes share the same ID. The segment

nodes are assigned IDs according to their activation rounds such that a node that was activated at round j is assigned the ID j .

Algorithm 1: K-TUPLE FULL WITHHOLDING.

```

Initialization
 $k = 4$  /* the segment size = 4, 8, 16, 32, 64 */
 $range\_stack = \{(1, k)\}$  // stack of ranges to check
 $range \leftarrow \emptyset$  // current range
/* segment process start */
 $node\_id = activation\_round$  // getting node id based on activation round
repeat
     $range \leftarrow Pop(range\_stack)$ 
    if  $node\_id$  in  $range$  and  $IsQueueNotEmpty(node\_id)$  then
        Broadcast()
        // transmitting a packet from the queue on the channel
    if  $feedback = silence$  then
        if  $range\_stack$  is empty then
            Terminate() // ending the segment processing
        else if  $feedback = packet$  then
            /* the node keeps broadcasting until the queue is empty */
            repeat
                Broadcast()
            until  $feedback = silence$ 
            Clear( $range\_stack$ )
            Push( $range\_stack, (1, k)$ )
        else
            /* collision case */
            if  $Size(range\_stack) = 2$  then
                /* the two nodes transmits all their packets consecutively */
                repeat
                    /* the first node transmitting */
                    Broadcast()
                    until queue is empty for the first node Wait_Silence()
                repeat
                    /* the second node transmitting */
                    Broadcast()
                    until queue is empty for the second node
                Wait_Silence()
                Clear( $range\_stack$ )
                Push( $range\_stack, Range((1, k))$ )
            else
                /* divide the range and store in the stack */
                Push( $Range(range\_stack, first\_half)$ )
                Push( $Range(range\_stack, second\_half)$ )
until segment terminates

```

During the segment processing phase, a binary search tree is performed to find a range of nodes with a single active node. The searching process begins from the initial range, in which all nodes with IDs in the range $[1, k]$ are given access to the channel. In turn, these nodes will attempt to transmit their packets in the current round. When the feedback is a collision, the search process divides the range into two halves, which are processed consecutively. Otherwise, when a node successfully transmits its packet, the node retains sole access to the channel, which is known as *withholding*. The node continuously transmits all packets from its queue for the next several rounds until the queue is empty. When silence is detected on the channel after the successful transmission of all packets in the

queue, the segment processing phase is reset so that the new search starts from the root node, in which all nodes in the range $[1, k]$ participate in the execution.

Figure 1. How the binary search works in the K-tuple full withholding (KTFW) when $k = 8$.

The algorithm employs a special processing case when the searching process reaches a range of size 2. When there is collision feedback, the algorithm provides the two nodes with consecutive access to the channel, which is first provided to the node with the lower ID to transmit all packets from its queue. The next node begins transmitting its packets when silence is detected. Finally, the segment processing is reset when the second node finishes transmitting all packets, and the second silence is detected. The segment terminates when silence is identified at the root node, which implies that all packets from all activated nodes have been transmitted successfully. Afterward, the algorithm continues processing the next segment, which consists of a new set of activated nodes. All nodes run the broadcast algorithm as long as they remain connected to the network, even when they are passive. Each node only participates in the segments in which it was activated, while the node stays silent during other segments.

Next, we describe the synchronization step which is performed by any new node joining the network. This step is required to maintain the synchronized execution of the KTFW algorithm between all nodes. Because all nodes have locally synchronous clocks, synchronization is performed through a local round number reset by identifying unique cases. In the synchronization step, two cases are considered, in which network activity is either present or not. In the case of network activity, a new node must identify the start of a segment before beginning the execution of the KTFW algorithm. In this step, the node keeps listening to feedback from the channel until a unique pattern is identified. This unique pattern is represented by two consecutive silence feedback, which represents the end of a segment. When a node identifies this pattern, it can then synchronize its execution of the KTFW algorithm with existing nodes in the network. Therefore, new nodes can join without having their clocks synchronized with existing nodes. Any new node can start communicating on the network after completing the synchronization step.

The second case for the synchronization step is when no network activity is present and the new node has at least one packet to transmit. In such cases, the new node waits for a maximum of k rounds before beginning the execution of the KTFW algorithm. The existing nodes synchronize their execution with the new node as soon as it starts processing a segment, represented by a packet being heard on the channel. When a collision is received as feedback, it implies that the existing nodes started processing a segment. In this scenario, the new node synchronizes its execution with the existing nodes and continues its execution of the KTFW algorithm accordingly. The synchronization step is performed exactly once by any new node, which allows them to communicate as long as they are connected. Because the synchronization step is performed once when a new node joins the network, it does not affect the network's performance.

Next, we present an example of how the KTFW algorithm works, assuming $k = 8$. After 8 rounds elapsed, the queues of the nodes are $\{1, 1, 2, 0, 0, 0, 0, 1\}$, and their IDs based on the activation round are $\{1, 2, 3, -, -, -, -, 8\}$. The nodes are referred to as $n_1, n_2, \dots, n_8$. The execution of the KTFW algorithm begins in round 9. All nodes with IDs in the range

(1 – 8) transmit their packets at this round. A collision is detected in the same round by all the nodes. Therefore, the range is divided into two halves: (1 – 4) and (5 – 8). The first half of nodes (1 – 4) transmit their packets at round 10, which results in a collision. Similarly, the range is divided into two halves: (1 – 2) and (3 – 4). Then, n_1 and n_2 transmit their packets in round 11, resulting in a collision. Because the range (1 – 2) consists of two nodes, the next rounds are used by both nodes to transmit their packets. Therefore, n_1 transmits its packet in round 12, and then after detecting silence in round 13, n_2 will transmit its packet. The nodes n_1 and n_2 become passive as they have transmitted all their packets. After detecting silence in round 15, the algorithm is reset by setting the search range to (1 – 8) in all active nodes. Next, a collision is found in round 16 because the range (1 – 8) contains two active nodes: n_3 and n_8 . Then, node n_3 transmits its two packets in rounds 17 and 18, and then it becomes passive. The algorithm is reset in round 19 because silence is detected. Afterward, node n_8 transmits its packet at round 20 because it is the only active node in the range (1 – 8). Finally, silence is detected in rounds 21 and 22, and the second silence implies the end of the segment since it is detected after the algorithm's reset.

Next, we highlight the novelty of the algorithm and describe how it is fundamentally different from the existing tree-based algorithms [37–39]. Although they share a common methodology of processing nodes for broadcast using a tree-based approach, they are different in two major ways: how the tree is constructed and how the nodes are processed for broadcasting. Typically, a tree is formed based on nodes with packets that must be broadcast for these algorithms. For the existing tree-based algorithms, a tree is constructed with nodes that participated in a collision. Therefore, there is no bound on the number of nodes in a formed tree. In contrast, in the KTFW algorithm, the tree comprises nodes that were activated during a segment consisting of k rounds, in which the maximum number of activated nodes is k .

After forming the trees, the nodes are processed to broadcast their packets while attempting to minimize collisions. In the existing tree-based algorithm, the processing of the tree uses a randomized approach where the nodes are subdivided through random coin flips. The KTFW algorithm uses a deterministic approach where nodes are subdivided through predetermined ranges of nodes. Clearly, the KTFW algorithm is inherently different from the existing tree-based broadcast algorithms.

5. Model of Injection

In this section, we describe the models used for injecting packets into nodes. The performance of the three broadcasting algorithms is evaluated using packet injection models that simulate network traffic activity [41,42]. Deterministic broadcasting algorithms are analyzed using an adversarial model of injection, in which injection is controlled by an adversary. One of the main advantages of using an adversarial model of injection over Poisson arrival of a packet is the absence of stochastic assumptions [43–46]. Two different traffic injection models are used to evaluate the performance of these broadcast algorithms. One of these models is based on the leaky bucket adversary [20,47], in which the adversary has full control over which nodes get injected with new packets. The LBIM possesses knowledge about the type of broadcasting algorithm being used. Based on this knowledge, the LBIM simulates worst-case network traffic such that injected packets make queues grow significantly or out of control as a result of the algorithm's failure to keep up with injected packets. This model is defined based on the parameters ρ and β , where ρ represents the injection rate, which is a real number such that $0 < \rho \leq 1.0$. The parameter β represents the maximum number of packets that can be injected in a single round, also known as the burstiness, which is an integer number such that $\beta \geq 1$. The second adversarial model of injection is based on the randomized adversary developed in [21]. The RIM is based on the LBIM, with an additional restriction on the power of the adversary. In this model, each node has an individual injection rate ρ_i such that $\sum_{i=1}^n \rho_i = 1.0$, where i is the node's number. The RIM simulates network traffic differing from that of the LBIM. The two traffic injection

models are also known as one-activating, as they can activate no more than one node in each round.

We studied the network traffic behavior of both types of adversarial models and compared them with real-world network traffic. The main objective of this comparison is to demonstrate that the network traffic behavior for the two adversarial models of injection resembles the network traffic behavior of real-world networks. The network traffic behavior is compared based on injection rates of packets among the nodes during a specified time period. A capture tool was developed in Python using the Scapy library to collect real-world network traffic [48]. This tool was used to capture wireless network traffic in several environments, including university campuses, hotels, coffee shops, and residential apartment complexes (RAC). For each environment, the traffic was captured for a period of 1–24 h and saved in pcap files. An analysis tool was also developed to process and analyze each pcap file for different time durations including 5, 10, and 15 minutes, which transforms the data into synchronous timing and converts each packet's source and destination addresses as a unique ID, beginning from 1. The transformed data was then saved in a CSV file, which was then analyzed for the number of active nodes and the injection rate of packets per node per unit of time.

For both the LBIM and the RIM, a simulation was developed in Java to model their injection behavior. Several simulations were run for different network sizes based on the number of nodes. The network traffic was recorded for each simulation as a CSV file. Charts for each simulation were developed that displayed similar network traffic behavior for each run regardless of the number of nodes in the network. For the RIM, the injection rate for each node was within 1% of any other node, which showed almost uniform distribution of injected packets (see Figure 2). As the RIM has the additional restriction of individual injection rates as well as randomization, this resulted in almost equal distribution of injected packets among the nodes.

Figure 2. packets injection rates for the randomized injection model (RIM) for a 10 min interval.

In comparison, for the LBIM, a single node had the highest injection rate, while all the other nodes had an almost 0% injection rate (see Figure 3). This situation is explained by how the LBIM works against a particular broadcast algorithm, for example, the Round Robin Withholding (RRW) broadcast algorithm. In the RRW algorithm, a single virtual token is passed in a round-robin fashion. The node holding the token will broadcast

all packets from its queue and then passes the token to the next node in the cycle when it detects silence. The LBIM follows a special injection strategy to make queues grow typically large, beginning by choosing a node as the start of the cycle. Then, the LBIM begins injecting packets into the node preceding the one holding the token. When the token has completed a cycle, the LBIM will switch its injection behavior such that it injects packets into the node following the one holding the token. Due to this switching, one node's queue becomes flooded with injected packets compared to all other nodes, explaining why one node has the highest injection rate.

Figure 3. Packets injection rates for the LBIM for a 10 min interval.

For real-world network traffic, several charts were generated using analysis tools. For environments such as the school library and a hotel, the charts show that a typically low number of nodes generated most of the network traffic, with a relatively higher injection rate compared to all other nodes (see Figure 4). This trend was observed for all interval durations of 5, 10, and 15 min. On the other hand, environments such as residential apartment complexes (RAC) exhibited behaviors more consistent with the RIM (see Figure 5). In this type of network traffic behavior, the injection rate of packets was distributed almost uniformly among the nodes.

To ensure that the observed traffic conforms to either of the traffic injection models, we calculated the distance between every collected traffic trace and both injection models. The distance between the two traffic traces is defined as the distance between their distributions of injection rates among nodes. To calculate this distribution, we classified nodes' injection rates into a predefined set of fixed-size bins, where each bin represents a range of injection rates. The probability of each bin was then calculated as the number of nodes in each bin divided by the total number of nodes.

We generated this probability distribution for every collected and simulated network traffic trace. To calculate the distance between these two distributions, we used a divergence metric. KL-divergence is a method used for measuring the distance between two probability distributions [49]. However, KL-divergence fails when compared probability distributions do not have absolute continuity, i.e., some bins are empty. Therefore, we used a modified version of KL-divergence with no requirement of absolute continuity to compute the

difference between these injection rates [50]. The modified KL-divergence is computed based on the following equation,

$$D(p, j) = \sum_{x \in X} \frac{p(x)}{(0.5 \times p(x) + 0.5 \times j(x))} \quad (1)$$

where p and j are the two compared probabilities. The probability p belongs to the main probability distribution that is being compared, which is the real-world network traffic. The probability j is obtained from the probability distribution of the simulated network traffic.

Figure 4. Packets injection rates of captured network traffic in a hotel.

Figure 5. Packets injection rates of captured network traffic in a residential apartment complex.

We computed the divergence between simulated traffic of both injection models and each collected traffic using the modified KL-divergence (see Table 1). The lower divergence value for each network environment is underlined. As shown in the table, networks such as the school's library or the school's department are more represented by the LBIM (have lower divergence from it), whereas the RAC environment is close to the RIM. For some network environments such as the hotel, depending on the interval duration, the network traffic could be simulated by either injection models. Therefore, both types of traffic injection models exhibited certain network traffic behavior that exists in real-world networks. Thus, we evaluate our new broadcast algorithm for both traffic injection models.

Table 1. KL-divergence of the RIM and the LBIM for different network environments.

Network Environment	KL-Divergence	
	RIM	LBIM
RAC 1	<u>0.368829</u> ⁱ	0.693147
RAC 2	<u>0.368829</u>	0.693147
RAC 3	<u>0.150489</u>	0.222243
School's Library 1	0.439448	<u>0.058563</u>
School's Library 2	0.398518	<u>0.057747</u>
School's Department 1	0.410021	<u>0.155151</u>
School's Department 2	0.437304	<u>0.153306</u>
School's Department 2	0.490143	<u>0.33193</u>
Coffe Shop	0.01894	<u>0.017405</u>
Hotel 1	0.201748	<u>0.09199</u>
Hotel 2	<u>0.263411</u>	0.296524
Hotel 3	0.693147	<u>0.11833</u>
Hotel 4	0.004971	<u>0.002968</u>
Hotel 5	<u>0.004933</u>	0.007417

ⁱ Underlined value represents the lower divergence.

6. Jamming Model

In this section, we describe the design of the network jamming model, which is called the randomized jammer. This entity, which is also known as a memoryless jammer, is defined based on the parameter $j \in [0.0, 1.0]$, which controls the jamming power. For any number of rounds τ , the channel is jammed for at most $j * \tau$ rounds. This model specifies the jamming effect on the channel, which results in network disruption. Thus, it is not associated with the number of jamming entities but with the collective jamming power of these entities.

This jamming model was chosen to assess the performance of the broadcast algorithms under both low and high jamming attacks. Furthermore, it can have more jamming power than intelligent jammers, which are typically designed for energy efficiency or to avoid detection. Therefore, this model can summarize the performance differences between the broadcast algorithms under jamming attacks with different levels of power and sophistication.

When the jammer attacks the channel in a specific round, there are three possible scenarios. In the first scenario, no legitimate node is transmitting, which would result in hearing the packet that was transmitted by the attacker node. This packet is detected as nonessential by all nodes and is not counted towards the throughput. The second scenario takes place when a single legitimate node transmits a packet on the channel. In such a case, the packet transmitted by the attacker node causes a collision on the channel, which prevents the packet from being successfully transmitted. The third scenario occurs

when two or more legitimate nodes transmit a packet on the channel. The transmission of another packet by the attacker does not have any effect on the resulting feedback, which is a collision. Following this event, the jammer starts jamming the network as soon as the execution begins, based on the jamming rate j .

7. Evaluation

In this section, we first describe the selected method for the evaluation, and then we present and discuss the results of the three broadcast algorithms. To compare our method with competing broadcast algorithms, we developed a network simulation tool to assess their performance under jamming attacks. Several network simulation tools can be used to assess the network's performance in various aspects [51,52]. Many of these tools can accurately predict the performance by providing a high level of details regarding the network's technical aspects. However, they can fail in predicting the performance of future network protocol and technology changes. Furthermore, using a simulation tool that implements all specifications of the networks' protocols complicates the analysis of the networks' performance. Thus, the developed simulation tool provides the needed level of detail to evaluate the network's performance for the three broadcast algorithms.

We developed a simulation tool that simulates a MAC network with a given number of nodes attached. The simulation models the behavior of multiple nodes communicating through a single shared channel. All communication aspects were implemented, from broadcasting packets to receiving feedback from the channel. The simulation operates with synchronous communication, in which time is divided into equal time slots, also known as rounds. Nodes could then transmit a packet on the channel in a single round or listen to its feedback. The simulation implements an abstraction of the MAC network without the various layers of the network protocols. This implementation allows us to investigate the effect of the broadcast algorithm on communication performance in isolation. Using the developed simulation, we analyzed and compared all algorithms under identical network scenarios for injection and jamming. A screenshot of the simulator is shown in Figure 6 where different parameters can be controlled including the number of nodes, the jamming rate and the injection model.

Figure 6. Screenshot of the developed multiple access channel (MAC) network simulator.

The traffic injection model was implemented based on the two types of injection models described in Section 5. The LBIM and RIM generate packets randomly based on ρ and β . For all simulations, the parameters ρ and β were set to fixed values. In this work, we focus on high network traffic scenarios, where the broadcast algorithm becomes one of the main factors affecting MAC network performance. The injection rate ρ was set to 1 to evaluate the performance under relatively high network traffic. The simulation was run for different burstiness values in the range [1–100], and the result indicated no effect on the performance for the three broadcast algorithms. Therefore, the burstiness was set to 20 in this evaluation.

In Figure 7, the standard deviation of the average packet latency is shown for 100 simulation runs. These results indicate that both the KTFW and QB algorithms have minor deviations from the average packet latency. In contrast, the BEB algorithm's standard deviation is in the range of [900–1500], which is due to randomization. These results are expected because the maximum window range is 1024 for the BEB algorithm.

Figure 7. The standard deviation of the average packet latency for the three broadcast algorithms for 100 simulation runs and $n \in [10-100]$.

The performance of the KTFW was evaluated for different k values from the set $\{4, 8, 16, 32, 64, 128, 256\}$ to find the optimal k value with the lowest average packet latency. The results in Figure 8 show the average packet latency of all k values under the LBIM. The result shows increasing average packet latency for all k values as the number of nodes increases. The average packet latency for $k = 4$ is the lowest among all k values, so the value k is set to 4 in this evaluation.

The broadcast algorithms are evaluated on different network scenarios to cover most types of networks based on the size and the jamming power. They are evaluated for both small and large network sizes and networks with low and high jamming power. Several simulations were run based on two parameters: the number of nodes and the jamming rate. The number of nodes represents the network size, which is denoted as n . MAC networks can have a large number of nodes; however, it is typical for them to have hundreds or fewer nodes [10,12,14]. Therefore, the value n is defined in the range [10–100], with a step size of 10. The jamming rate is chosen from the range [0.0–1.0], with a step size of 0.05. These simulations were run for the two injection models of the LBIM and RIM. The three broadcasting algorithms were compared based on two performance measures: the throughput and the average packet latency. The throughput of a broadcasting algorithm is

defined as the number of successfully transmitted packets divided by the total number of injected packets during the whole execution. Packet latency is computed as the number of rounds a packet waits in a queue before it is successfully transmitted.

Figure 8. Comparing the average packet latency for different k values of the KTFW algorithm and all $n \in [10-100]$.

For each configuration, the performance results were computed by averaging the results of 100 simulations. The simulation was individually run for each broadcast algorithm using the two injection models (the LBIM and the RIM). The throughput and the average packet latency results were recorded and then used to generate charts to compare the performances of the three broadcast algorithms.

Using the simulation, we compared the performances of the three broadcast algorithms for all jamming rates in the range $[0.0-1.0]$, with a step of 0.05. As seen in Figures 9 and 10, the KTFW has a higher throughput than the BEB algorithm. For $n = 10$, the BEB throughput is 0.95 while the KTFW achieves the perfect throughput of 1. Moreover, as the number of nodes increases, the superiority of the KTFW over the BEB in terms of throughput increases. As the figures show, the BEB throughput decreases by around 0.05 for every increment in the number of nodes. This degradation in the throughput is due to the inherited randomness of the BEB in which a node chooses a random backoff time before retransmitting a packet. The probability of collision increases in the channel as the number of nodes grows, which in turn affects the throughput. On the other hand, the KTFW's throughput is not affected for $n \geq 20$. The difference in the throughput becomes as large as 0.5 when $n = 100$ and $j = 0$, and it diminishes as the jamming rate approaches 1 (see Figure 10).

In general, the results show that the throughput of the KTFW is not affected in both small and large network sizes. In contrast, the throughput of the BEB degrades as the network grows larger due to how the algorithm behaves. As more nodes contend for access to the channel, they typically wait for longer durations because a packet can experience multiple collisions before being successfully transmitted. In the KTFW, the throughput is not affected because all nodes follow the same process to transmit all injected packets even when the number of nodes increases. These packets are processed in segments that consist of at most k active nodes. As there is no random access to the channel, the KTFW is able to maintain the same throughput for a larger number of nodes.

Figure 9. Comparing the throughput of the KTFW and the BEB algorithms for all jamming rates and $n \in [10-50]$.

Figure 10. Comparing the throughput of the KTFW and the BEB algorithms for all jamming rates and $n \in [60-100]$.

The throughput results for the QB algorithm with no jamming are the same for both types of injection models and is equal to 1. For any $j \geq 0$, the throughput value becomes 0 (see Figure 9). The main reason for this huge degradation in throughput performance is the reliance of the QB on control information for managing access to the channel. As the algorithm is adaptive, additional information is attached to packets, which are used to control access to the channel based on queue paradigm. However, failure in transmission of a single critical piece of information, such as the over bit flag, could cause the nodes to

become stuck in line. As a result, with even a jamming rate of 0.05, all nodes remain in a listening state, waiting for the bit flag to be received on the channel. The QB algorithm is not designed to handle special scenarios such as lost packets due to jamming attacks.

We compared the average packet latency results for all three broadcast algorithms with no jamming and $n = [10-100]$ (see Figure 11). The results are shown for both injection model types, revealing that the average packet latency increases with the number of nodes. For the BEB, the average packet latency experiences a much larger increase than both the KTFW and the QB algorithms. These results also show that both the QB and the KTFW algorithms have significantly lower average packet latency than the BEB algorithm. For the RIM, the QB and the KTFW algorithms have almost identical average packet latency for all values of n . In contrast, the QB performs slightly better than the KTFW algorithm for all $n > 50$, while they perform identically for all $n \leq 50$.

(a) LBIM

(b) RIM

Figure 11. Comparing the average packet latency for the three algorithms (QB, KTFW, and BEB) for all $n \in [10-100]$.

The results for average packet latency under jamming indicates lower average packet latency for the KTFW in all cases except for when $j = [0.15-0.4]$ and $n = 10$, in which case both algorithms have similar average packet latency (see Figure 12). In contrast, for the LBIM, the KTFW outperforms other algorithms in all cases. Results also show that the number of nodes has a huge impact on the average packet latency for the BEB algorithm, which increases significantly as n grows. The reason for this increase in average packet

latency is that the jammer will cause most retransmissions of packets to fail. Successive failure of retransmission implies a higher backoff window range, which leads to longer waiting times for packets. In addition, the average packet latency significantly increases for the BEB algorithm as the jamming rate increases. In comparison, the average packet latency for the KTFW algorithm has a slower growth rate than the BEB algorithm, which shows its resilience to jamming attacks.

In Figure 12, an unusual result can be observed where the average packet latency for the KTFW algorithm with $n = 10$ (KTFW 10) becomes larger than with $n = 50$ (KTFW 50). The transmission of more packets fails due to the increased jamming power, which increases the segment processing duration. Furthermore, the injection model has a higher probability of injecting packets into the current segment nodes when $n = 10$. Accordingly, the segment processing duration is typically longer for the KTFW 10 than for the KTFW 50 under high jamming rates. This produces longer delays for packets in the other remaining nodes that are not part of the current segment.

In summary, the KTFW algorithm results demonstrate a significant advantage in the two performance measures compared with the standard BEB and QB algorithms. Although both the KTFW and BEB algorithms recorded nearly identical throughputs and average packet latencies when the network is small (e.g., $n = 10$), the KTFW algorithm becomes significantly more efficient as the network size increases. The KTFW algorithm outperforms the BEB algorithm while maintaining a lower average packet latency, whereas the BEB suffers from a significant increase in the average packet latency, especially for high jamming rates. However, the effect on the average packet latency for the KTFW algorithm is fairly marginal for any number of nodes, demonstrating its resilience to jamming attacks compared to the BEB algorithm. In addition, when no jamming occurs, the QB and the KTFW algorithms exhibit almost identical performance in terms of average packet latency. In contrast, the QB algorithm has a slightly lower average packet latency for larger networks in certain traffic scenarios (e.g., the RIM). Nevertheless, the KTFW algorithm performs better than the QB algorithm when considering networks with jamming attacks.

Figure 12. Comparing the average packet latency for the BEB and KTFW algorithms for all jamming rates and $n \in [10, 50, 100]$.

8. Conclusions

In this paper, we proposed a novel broadcast algorithm, called the KTFW algorithm, that is more resilient to jamming attacks and is scalable for larger networks. We evaluated the performance of this algorithm through simulations and compared it with the standard BEB and QB algorithms. For each algorithm, the performance was measured in terms of the throughput and average packet latency for various network sizes, traffic injection models, and jamming rates. The results display a higher performance for the KTFW algorithm than for the BEB algorithm against jamming attacks, especially for larger networks. In contrast, the QB algorithm fails to maintain network performance when jamming attacks occur. The resilience of the KTFW algorithm to jamming attacks is reflected in the average packet latency results, where jamming attacks have a marginally lower effect on the average packet latency of the KTFW algorithm than that of the BEB algorithm. The new proposed algorithm could improve the performance, reliability, and scalability of MAC networks. In the future, we will work on investigating methods for implementing the KTFW algorithm in WLANs and replacing the BEB algorithm. Furthermore, we plan to study the performance of the KTFW algorithm for larger network sizes and other types of jamming models, including intelligent jammers. Studying and analyzing the algorithm fairness compared to the other algorithms is another possible direction of the work. It will be interesting to determine how the algorithm performs under other types of traffic injection models.

Author Contributions: Conceptualization, B.A.A.; methodology, B.A.A.; simulation implementation, B.A.A.; analysis, B.A.A.; manuscript writing original draft, B.A.A.; writing—review and editing, B.A.A. and J.H.J.; Supervision, J.H.J. Both the authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Lin, G.; Noubir, G. On link layer denial of service in data wireless LANs. *Wirel. Commun. Mob. Comput.* **2005**, *5*, 273–284. [\[CrossRef\]](#)
2. Bicakci, K.; Tavli, B. Denial-of-Service attacks and countermeasures in IEEE 802.11 wireless networks. *Comput. Stand. Interfaces* **2009**, *31*, 931–941. [\[CrossRef\]](#)
3. Akyildiz, I.F.; Wang, X. A survey on wireless mesh networks. *IEEE Commun. Mag.* **2005**, *43*, S23–S30. [\[CrossRef\]](#)
4. Siddiqui, M.S. Security issues in wireless mesh networks. In Proceedings of the 2007 International Conference on Multimedia and Ubiquitous Engineering (MUE'07), Seoul, Korea, 26–28 April 2007; pp. 717–722.
5. Chan, A.; Liu, X.; Noubir, G.; Thapa, B. Broadcast control channel jamming: Resilience and identification of traitors. In Proceedings of the 2007 IEEE International Symposium on Information Theory, Nice, France, 24–29 June 2007; pp. 2496–2500.
6. Xu, W.; Trappe, W.; Zhang, Y. Channel surfing: Defending wireless sensor networks from interference. In Proceedings of the 2007 6th International Symposium on Information Processing in Sensor Networks, Cambridge, MA, USA, 25–27 April 2007; pp. 499–508.
7. Xu, W.; Trappe, W.; Zhang, Y. Anti-jamming timing channels for wireless networks. In Proceedings of the First ACM Conference on Wireless Network Security, Alexandria, VA, USA, 31 March–2 April 2008; pp. 203–213.
8. Li, C.; Dai, H.; Xiao, L.; Ning, P. Communication efficiency of anti-jamming broadcast in large-scale multi-channel wireless networks. *IEEE Trans. Signal Process.* **2012**, *60*, 5281–5292. [\[CrossRef\]](#)
9. Grover, K.; Lim, A.; Yang, Q. Jamming and anti-jamming techniques in wireless networks: A survey. *Int. J. Ad Hoc Ubiquitous Comput.* **2014**, *17*, 197–215. [\[CrossRef\]](#)
10. Murty, R.; Padhye, J.; Chandra, R.; Wolman, A.; Zill, B. Designing High Performance Enterprise Wi-Fi Networks. In NSDI'08, Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation, San Francisco, CA, USA, 16–18 April 2008; USENIX Association: Berkeley, CA, USA, 2008; pp. 73–88.
11. Gupta, A.; Min, J.; Rhee, I. WiFox: Scaling WiFi Performance for Large Audience Environments. In CoNEXT '12, Proceedings of the Eighth Conference on Emerging Networking EXperiments and Technologies, Côte d'Azur, France, 10–13 December 2012; ACM: New York, NY, USA, 2012.
12. Maity, M.; Raman, B.; Vutukuru, M. TCP Download Performance in Dense WiFi Scenarios: Analysis and Solution. *IEEE Trans. Mob. Comput.* **2017**, *16*, 213–227. [\[CrossRef\]](#)
13. Liu, B.; Zhu, Q.; Tan, W.; Zhu, H. Congestion-optimal WIFI offloading with user mobility management in smart communications. *Wirel. Commun. Mob. Comput.* **2018**, *2018*, 9297536. [\[CrossRef\]](#)

14. Ergin, M.A.; Ramachandran, K.; Gruteser, M. An experimental study of inter-cell interference effects on system performance in unplanned wireless LAN deployments. *Comput. Netw.* **2008**, *52*, 2728–2744. [\[CrossRef\]](#)
15. Raghavendra, R.; Belding, E.M.; Papagiannaki, K.; Almeroth, K.C. Unwanted Link Layer Traffic in Large IEEE 802.11 Wireless Networks. *IEEE Trans. Mob. Comput.* **2010**, *9*, 1212–1225. [\[CrossRef\]](#)
16. Peng, J.; Cheng, L.; Sikdar, B. A Wireless MAC Protocol with Collision Detection. *IEEE Trans. Mob. Comput.* **2007**, *6*, 1357–1369. [\[CrossRef\]](#)
17. Sen, S.; Choudhury, R.R.; Nelakuditi, S. CSMA/CN: Carrier sense multiple access with collision notification. *IEEE/ACM Trans. Netw.* **2011**, *20*, 544–556. [\[CrossRef\]](#)
18. Alassery, F.; Ahmed, W.K.; Sarraf, M.; Lawrence, V. A Novel Low-Complexity Low-Latency Power Efficient Collision Detection Algorithm for Wireless Sensor Networks. *Wirel. Sens. Netw.* **2015**, *7*, 43. [\[CrossRef\]](#)
19. Anantharamu, L.; Chlebus, B.S. Broadcasting in ad hoc multiple access channels. *Theor. Comput. Sci.* **2015**, *584*, 155–176. [\[CrossRef\]](#)
20. Chlebus, B.S.; Kowalski, D.R.; Rokicki, M.A. Maximum throughput of multiple access channels in adversarial environments. *Distrib. Comput.* **2009**, *22*, 93–116. [\[CrossRef\]](#)
21. Aldawsari, B.A.; Chlebus, B.S.; Kowalski, D.R. Broadcasting on Adversarial Multiple Access Channels. In Proceedings of the 18th IEEE International Symposium on Network Computing and Applications (NCA 2019), Cambridge, MA, USA, 26–28 September 2019.
22. Bellardo, J.; Savage, S. 802.11 Denial-of-Service Attacks: Real Vulnerabilities and Practical Solutions. In Proceedings of the USENIX Security Symposium, Washington, DC, USA, 4–8 August 2003; Volume 12, p. 2.
23. Negi, R.; Perrig, A. *Jamming Analysis of MAC Protocols*; Technical Report, Carnegie Mellon Technical Memo; Carnegie Mellon University: Pittsburgh, PA, USA, 2003.
24. Wu, B.; Chen, J.; Wu, J.; Cardei, M. A survey of attacks and countermeasures in mobile ad hoc networks. In *Wireless Network Security*; Springer: Boston, MA, USA, 2007; pp. 103–135.
25. Gummadi, R.; Wetherall, D.; Greenstein, B.; Seshan, S. Understanding and Mitigating the Impact of RF Interference on 802.11 Networks. In *SIGCOMM '07, Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Kyoto, Japan, 27–31 August 2007*; Association for Computing Machinery: New York, NY, USA, 2007; pp. 385–396.
26. Xu, W.; Trappe, W.; Zhang, Y.; Wood, T. The Feasibility of Launching and Detecting Jamming Attacks in Wireless Networks. In *MobiHoc'05, Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing, Champaign, IL, USA, 25–28 May 2005*; Association for Computing Machinery: New York, NY, USA, 2005; pp. 46–57.
27. Bayraktaroglu, E.; King, C.; Liu, X.; Noubir, G.; Rajaraman, R.; Thapa, B. Performance of IEEE 802.11 under jamming. *Mob. Netw. Appl.* **2013**, *18*, 678–696. [\[CrossRef\]](#)
28. Xu, W.; Ma, K.; Trappe, W.; Zhang, Y. Jamming sensor networks: Attack and defense strategies. *IEEE Netw.* **2006**, *20*, 41–47.
29. Thuente, D.; Acharya, M. Intelligent jamming in wireless networks with applications to 802.11 b and other networks. In Proceedings of the MILCOM, Washington, DC, USA, 23–25 October 2006; Volume 6, p. 100.
30. Xu, W.; Wood, T.; Trappe, W.; Zhang, Y. Channel surfing and spatial retreats: Defenses against wireless denial of service. In Proceedings of the 3rd ACM Workshop on Wireless Security, Philadelphia, PA, USA, 1 October 2004; pp. 80–89.
31. Jiang, S.; Xue, Y. Optimal wireless network restoration under jamming attack. In Proceedings of the 2009 18th International Conference on Computer Communications and Networks, San Francisco, CA, USA, 3–6 August 2009; pp. 1–6.
32. Dou, Z.; Si, G.; Lin, Y.; Wang, M. An adaptive resource allocation model with anti-jamming in IoT network. *IEEE Access* **2019**, *7*, 93250–93258. [\[CrossRef\]](#)
33. Tang, X.; Ren, P.; Han, Z. Jamming mitigation via hierarchical security game for IoT communications. *IEEE Access* **2018**, *6*, 5766–5779. [\[CrossRef\]](#)
34. Tseng, S.M.; Chen, Y.F.; Chiu, P.H.; Chi, H.C. Jamming resilient cross-layer resource allocation in uplink HARQ-based SIMO OFDMA video transmission systems. *IEEE Access* **2017**, *5*, 24908–24919. [\[CrossRef\]](#)
35. Abramson, N. The Aloha System: Another Alternative for Computer Communications. In *AFIPS '70 (Fall): Proceedings of the November 17–19, 1970, Fall Joint Computer Conference*; ACM: New York, NY, USA, 1970; pp. 281–285.
36. Roberts, L.G. ALOHA Packet System with and without Slots and Capture. *SIGCOMM Comput. Commun. Rev.* **1975**, *5*, 28–42. [\[CrossRef\]](#)
37. Capetanakis, J. Tree algorithms for packet broadcast channels. *IEEE Trans. Inf. Theory* **1979**, *25*, 505–515. [\[CrossRef\]](#)
38. Hayes, J. An Adaptive Technique for Local Distribution. *IEEE Trans. Commun.* **1978**, *26*, 1178–1186. [\[CrossRef\]](#)
39. Tsybakov, B.; Mikhailov, V. Free Synchronous Packet Access in a Broadcast Channel with Feedback. *Probl. Inform. Transm.* **1978**, *14*, 259–280.
40. Metcalfe, R.M.; Boggs, D.R. Ethernet: Distributed packet switching for local computer networks. *Commun. ACM* **1976**, *19*, 395–404. [\[CrossRef\]](#)
41. Aldous, D. Ultimate instability of exponential back-off protocol for acknowledgment-based transmission control of random access communication channels. *IEEE Trans. Inf. Theory* **1987**, *33*, 219–223. [\[CrossRef\]](#)
42. Kelly, F.P. Stochastic models of computer communication systems. *J. R. Stat. Soc. Ser. B Methodol.* **1985**, *47*, 379–395. [\[CrossRef\]](#)
43. Håstad, J.; Leighton, F.T.; Rogoff, B. Analysis of Backoff Protocols for Multiple Access Channels. *SIAM J. Comput.* **1996**, *25*, 740–774. [\[CrossRef\]](#)
44. Raghavan, P.; Upfal, E. Stochastic contention resolution with short delays. *SIAM J. Comput.* **1998**, *28*, 709–719. [\[CrossRef\]](#)

45. Borodin, A.; Kleinberg, J.M.; Raghavan, P.; Sudan, M.; Williamson, D.P. Adversarial queuing theory. *J. ACM* **2001**, *48*, 13–38. [[CrossRef](#)]
46. Bender, M.A.; Farach-Colton, M.; He, S.; Kuszmaul, B.C.; Leiserson, C.E. Adversarial Contention Resolution for Simple Channels. In *SPAA '05, Proceedings of the Seventeenth Annual ACM Symposium on Parallelism in Algorithms and Architectures, Las Vegas, NV, USA, 18–20 July 2005*; ACM: New York, NY, USA, 2005; pp. 325–332.
47. Andrews, M.; Awerbuch, B.; Fernández, A.; Leighton, F.T.; Liu, Z.; Kleinberg, J.M. Universal-stability results and performance bounds for greedy contention-resolution protocols. *J. ACM* **2001**, *48*, 39–69. [[CrossRef](#)]
48. Open Source. Scapy Packet Manipulation Tool (v2.4). 2020. Available online: <https://scapy.net/> (accessed on 5 January 2020).
49. Kullback, S.; Leibler, R.A. On Information and Sufficiency. *Ann. Math. Statist.* **1951**, *22*, 79–86. [[CrossRef](#)]
50. Lin, J. Divergence measures based on the Shannon entropy. *IEEE Trans. Inf. Theory* **1991**, *37*, 145–151. [[CrossRef](#)]
51. Heidemann, J.; Bulusu, N.; Elson, J.; Intanagonwiwat, C.; Lan, K.C.; Xu, Y.; Ye, W.; Estrin, D.; Govindan, R. Effects of detail in wireless network simulation. In *Proceedings of the SCS multiconference on Distributed Simulation, Phoenix, AZ, USA, 17 January 2001*; pp. 3–11.
52. Rampfl, S. Network simulation and its limitations. In *Proceeding Zum Seminar Future Internet (FI), Innovative Internet Technologien und Mobilkommunikation (IITM) und Autonomous Communication Networks (ACN)*; Citeseer: University Park, PA, USA, 2013; Volume 57.